

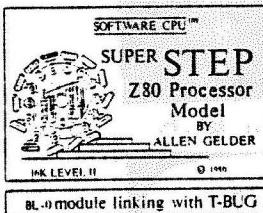


ALLEN GELDER & CO.
microcomputer software

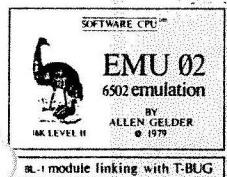
Box 11721 Main Post Office

San Francisco, CA 94101

Software CPU™

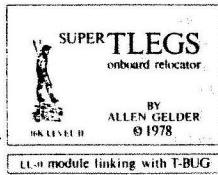


SUPER STEP: Everything that was left to your imagination is now brought to the screen! Namely: before/after Z80 Processor Models animated in response to a disassembled listing plus an Intelligent RAM window that selects memory environments to show you. Single-step or TRACE with background/foreground breakpointing, variable speed control, keyboard interrupt, dynamic SKIP key and more. 36 key functions service the display or help you do local editing, plus faster tape I/O, relocatability. A Z80 Software CPU. 16K Level II, TBUG required. No. BL-0 Super STEP \$19.95



EMU 02: How to have a 6502 without having a 6502! Actually two distinct programs in one; a powerful Cross-debugger with before/after 6502 CPU Models and stack for single-stepping or TRACE, and a FAST interpretive Cross-translator that will run 6502 machine code programs with realism. Single-step mode and TRACE mode both disassemble scrolling locations into standard 6502 mnemonic forms. 4-speed TRACE opens a keyboard scan port for user interaction with 6502 program material. Paging initialized in virtual address space. You can write, debug and execute 6502 machine language programs on your TRS-80, communicate with Apple, PET. And their owners! 16K Level II, TBUG required. No. BL-1 EMU 02 \$24.95

TBUG Accessories



SUPER TLEGS: Onboard relocator for TBUG. Lets TBUG move out of the way of intersecting programs, so no more revolting wipeouts by coincidence. And not only total address space access, but the ability to populate RAM with parallel independent TBUGs. So your TBUG can move to survive and replicate. Also will independently relocate Super STEP No. BL-0 and TSTEP, No. LL-1. 16K Level II, TBUG required. No LL-0 Super TLEGS \$9.95



NEW, IMPORTED FROM ENGLAND

ACCEL: Compiler for Level II BASIC. Compiles INTEGER subset with only minor constraints (no dynamic redefinition of names, statically associated FOR,NEXT loops) to fast Z80 machine code. Performance of the compiled program can be spectacularly improved in speed while size remains reasonable due to extensive use of ROM routines. ACCEL itself is of small size and self relocating, so even 16K machines can utilize the powerful compilation process. The compiled BASIC program requires only 256 bytes of ACCEL run-time routines. With no royalties on the derived code (except inclusion of copyright notice), the ACCEL produced speed improvements are available to the commercial software writer for production work. It's like having a 100 mhz clock! An extremely interesting and innovative program from Southern Software of England. ACCEL Compiler for Level II BASIC \$44.95

* Now we have ACCEL2 - - - Compiles Disk BASIC, all variable types
\$88.95

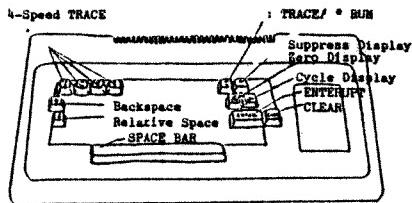


Fig. 1. Implicit keypad under EMU 02.

T-MQ Display Assembly		Programmed Address	Location and Function
		REF#	APP#
L100 A9	LDC PWD	A 02	A 02
L101 A0	ADC PWD	I 00	I 00
L103 00	ADC PWD	I 00	I 00
L104 FF	INT 0000	I 0000	I 0000
L105 FF	INT 0000	I 0000	I 0000
P 20	P 20	P 20	P 20
PL 00	PL 00	PL 00	PL 00
PL VC	PL VC	PL VC	PL VC
PL NC	PL NC	PL NC	PL NC
P100 FF	P100 FF	P100 FF	P100 FF
P101 FF	P101 FF	P101 FF	P101 FF
P102 FF	P102 FF	P102 FF	P102 FF
P103 FF	P103 FF	P103 FF	P103 FF
P105 FF	P105 FF	P105 FF	P105 FF

Fig. 2. Display under EMU 02

3.

Fig. 1. A page from the 16 page booklet



Fig. 2. What you get for \$24.95

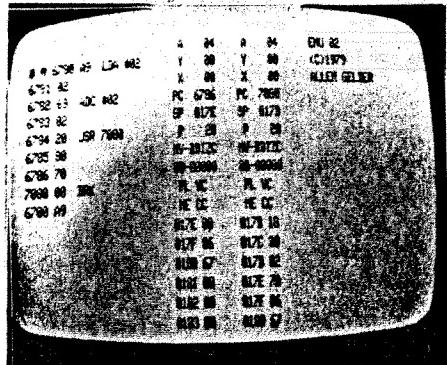


Fig. 3. Screen photo

EMU 02 6502 Emulation for TBUG is a software 6502 Cross-Interpreter for the Level II 16K TRS-80. Its size is approximately 1A00H bytes, plus 100H each bytes for page zero and page one, as initialized in virtual address space. 13 key functions service the display and function mode. The 6502 Processor Models (see Fig. 3) are animated in the Single-step or 4 speed TRACE modes. The fast translator mode executes 6502 code at a reasonable fraction of 6502 speed (some instructions over 6% of actual 6502 hardware speed). A keyboard scan port allows keyboard interaction. A 6502 Software CPU. \$24.95

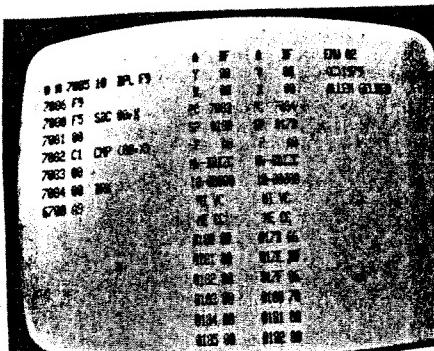


Fig. 4a Note the instruction sequence.

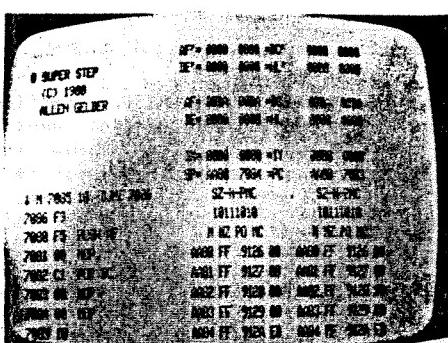


Fig. 4b It's all in how you look at it...

Control Options

If you have a large program to compile for the first time, especially if it's one you did not write yourself, then you may start by minimising the compilation, and then, when the compiled program is running, work up to full compilation by removing or localising the options. The options are embedded in the BASIC program using REM statements.

REM NOEXPR/EXPR will inhibit compilation of expressions within a bracketed critical section, and help contain code growth.

REM LINE will force generation of line numbers for error diagnosis.

REM NOARRAY will suppress compilation of array structures, making compatible adjustable bound arrays, e.g. INPUT N:DIM A(N).

	<u>INTEGER</u>	<u>SINGLE</u>	<u>DOUBLE</u>	<u>STRING</u>
Assignment (LET)	115	3.3	3.4	7.6
Array Reference (1-dim)	35	78	66	34.5
AND, OR	41	2.5	2.0	
Compare (=, etc.)	30	1.6	1.4	4.2
Add, Subtract, Concat.	47	2.0	1.5	4.9
Multiply (*)	3.3	2.0	1.5	
Divide (/)	2.0	2.0	1.02	
Reference to a constant	69	65	54	2.1
FOR with NEXT	15			
POKE	82	4.6	3.6	
SET or RESET	6.7	3.1	2.6	
IF THEN ELSE	11.1	3.0	2.3	7.6
ON expression GOTO	15.8	3.2	2.8	
<u>Functions</u>				
VARPTR	33	47	47	44
USR	11.2	3.7	2.8	
POINT	6.9	3.0	2.5	
PEEK	52	4.4	3.5	
LEN				43
MID\$				4.1
LEFT\$				3.0
RIGHT\$				2.8
CHR\$				4.7
ASC				30
CVI				28
<u>Flow of Control</u>				
GOSUB with RETURN	137			
GOTO	204			

Ratio of execution speeds for ACCEL2 against interpreter.

Selling Compiled Programs.

The core-image tape contains the ACCEL or ACCEL2 run-time routines that interface to the BASIC environment, and selling such tapes involves the resale of part of a Southern Software product. However it is too small a part to justify collecting royalties, and so the implicit resale will be ignored by Southern Software provided:-

- 1) The program is sold in its cassette form, not on disk.
- 2) No part of the compile-time routine is copied or resold.
- 3) An acknowledgment is given to Southern Software in the program documentation.

The following programs were compared for both speed and size, before and after compilation. For consistency of measurement the programs had no REMARKS and no keyboard input. The first example, the SORT, is instructive because it is possible to run exactly the same program, (with equivalent data values) against all four data types. ACCEL shows up badly on this example which is entirely concerned with shuffling array values. However it is possible to re-code the same example using PEEK and POKE, rather than arrays, to optimise its performance under ACCEL, and this is shown for comparison.

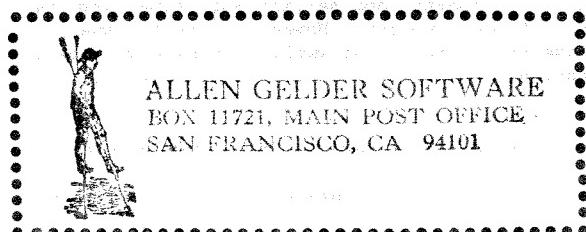
Sizes are in bytes, times in seconds. 'Gain' is the ratio of speed when compiled, to original speed.

Program	Uncompiled		ACCEL2					ACCEL			
	Size	Time	Size	Time	Gain	Compile Time	Size	Time	Gain	Compile Time	
Sort(INTEGER)	714	43.2	1230	1.8	24	4	937	34.4	1.3	3	
Sort(SINGLE)	714	43.2	1509	8.2	5.3	5	932	35.4	1.2	3	
Sort(DOUBLE)	714	46.8	1923	11.4	4.1	7	932	38.9	1.2	3	
Sort(STRING)	716	39.2	1391	4.3	9.1	5	932	32.4	1.2	3	
Sort(PEEK,POKE)	913	(216)					1276	5.7	7.6	7	
Screen Graphics	323	496	519	23	21.6	1	487	23	21.6	1	
Disk Dump	691	30.1	1316	10.3	2.9	4					
Income Tax	1184	39	2154	21	1.9	10	1381	37	1.1	5	
Game of LIFE	503	30	942	.8	39	3	939	.8	39	2	
Blackjack	3173	91	7380	32	2.8	115	5524	57	1.6	86	
Mann-Whitney (Statistics)	1914	15.5	3212	3.1	5.0	24	1960	15.5	1.0	14	

Restrictions.

1. No redefinition of meaning of names.
E.g. DEFSNG I : I = 1 : DEFINT I : I = 1 is disallowed.
2. Programs must be properly structured.
Each FOR-NEXT loop must be properly nested and uniquely terminated. Do not code e.g.
10 FOR I = 1 to 10
20 IF I = 5 THEN NEXT.
30 PRINT I : NEXT.
3. Behaviour of error conditions is not necessarily compatible. DATA-dependent errors, such as OVERFLOW or function argument out-of-range, are not necessarily diagnosed. The current line number (used in diagnosis, error handling, and in trace) is not accurately maintained.
4. Editing is not possible on the compiled program. The commands AUTO, CLOAD?, CSAVE, DELETE, EDIT, SAVE and MERGE are not meaningful and may not be used in a compiled program. NEW, LOAD or CLOAD must be used to reset the machine to its normal state.

Available in the US and Canada from:



SOUTHERN SOFTWAREACCEL and ACCEL2 COMPILERS for TRS BASIC

- * Have you ever wished your programs would run faster?
- * Do you have ideas for saleable programs you could implement, if only you had the time and knowledge to write machine-code?
- * Have you often wondered whether you should have bought a micro with a built in PASCAL compiler?
- * Why is it your one-megacycle CPU seems incapable of doing more than 500 additions per second?
- * Are your thumbs sore from sitting there, twiddling?

The remedy is simple: Get yourself a BASIC compiler from SOUTHERN SOFTWARE.

ACCEL	f19.95	(\$44.95)	2816 bytes	Level 2 BASIC
ACCEL2	f39.95	(\$88.95)	5120 bytes	Full Disk BASIC

ACCEL and ACCEL2 are versions of the same product. They will compile a BASIC 'source' program into an 'object' program which is compatible in function with the original, except that it runs faster. Performance improvements that can be achieved vary from spectacular (20 to 30 times) to modest (a few percent). Measured examples are given later. Both ACCEL and ACCEL2 will give outstanding improvements on programs of logic, such as games, music synthesis, screen graphics, searching algorithms, etc., while ACCEL2 will give valuable gains, 4 to 5 times, for string-handling programs. Neither will help programs that are entirely limited by I/O (disk, printer, tape, or keyboard).

ACCEL2 is a direct extension of ACCEL. It handles the full Disk BASIC, whereas ACCEL is limited to level 2. ACCEL2 will also produce performance improvements that ACCEL will not, notably in STRING handling, in SINGLE and DOUBLE arithmetic, and in manipulation of one-dimensional fixed-bound arrays. You'll need 16K of memory (or more) to run ACCEL satisfactory, and 32K of memory for ACCEL2 with Disk BASIC. If you want to use ACCEL2 on level 2 (non-Disk) then 16K is viable.

Southern Software programs are distributed on cassette and are self-relocating. When you load the original tape you can choose to locate the program anywhere in memory. This means you can load Southern Software programs concurrently with other Southern Software programs or with programs from other vendors, and you can upgrade your memory without problems.

The relocated programs can be saved on disk using TRSDOS DUMP, or on tape using TRS TBUG, or Southern Software TSAVE, for subsequent direct loading.

The Mechanics of Compilation.

Using ACCEL or ACCEL2, you get the advantages of both interpretation and compilation. Programs are built, modified and debugged using the BASIC editor/interpreter in the usual way. When correct, the program is compiled to get improved execution speed. The source form of the program (in BASIC) can be saved and reloaded in the normal way, using SAVE and LOAD, or CSAVE and CLOAD. But the compiled program no longer has the structure of a normal source program, and it cannot be edited or modified in any way, nor can it be saved and loaded with normal commands. To save a compiled program on tape you will need the separate Southern Software utility TSAVE (price f4.95 or \$9.95). The core-image file produced can then be reloaded using the SYSTEM command. With ACCEL2, under TRSDOS, you can save the compiled program core-image on disk, and reload it, using routines that are built into the compiler.

Capabilities of the Compilers.

The result of compilation is a program which is a mixture of BASIC statements and directly executing Z80 machine instructions. The run-time routines provided with ACCEL and ACCEL2 give control to the interpreter when a BASIC statement is to be executed, and they also ensure that the variable values accessed by the interpreter and the compiled code are consistent. The rule is that if a statement contains any operation that the compiler cannot convert to machine-code, then the whole statement is left in interpretive form. So if you are considering sale of your programs, you should allocate some time to tuning the program to the capabilities of the compiler, which are of course directly tied to the capabilities of the Z80 CPU. Any item not included in the following list, e.g. SIN (X) or X^AY, will inhibit the optimisation as machine-code of the statement in which it appears, but will not prevent correct execution.

Translation to Machine-Code.

Function	ACCEL	ACCEL2
GOTO, GOSUB, RETURN, RESTORE, IF, THEN, ELSE, CLEAR, ON,	Always	Always
LET, (Assignment), POKE, SET, RESET, POINT, PEEK, USR, VARPTR, +, -, AND, OR, NOT, = and all compares	Integer arguments only	All data types
*, / (multiply, divide)	No	All data types
Constants, e.g. 123, 12.3, "123"	Integers (-32768 to 32767)	All types
LEN, MID\$, LEFT\$, RIGHTS\$, CHR\$, ASC, CVI	No	All data types
One-dimensional, fixed-bound arrays	No	All data types

Preresolution of Names and Line Numbers.

The BASIC interpreter finds the location of each variable by a sequential execution. By contrast, the compiler allocates storage for each variable once during compilation, and replaces each reference to that variable by a direct machine address. Similarly each line reference in GOTO or GOSUB is translated to a branch address, whereas the BASIC interpreter searches sequentially through the program to find each target line. The longer the program, and the more variables it contains, then the greater the performance improvement that results from compilation.

Program Size.

The compiled machine instructions normally occupy more space than the BASIC source statements they replace. To counteract this the compiler removes REMARKS from the program, so its final size may be larger or smaller.

Space required by the compiler itself:-

	Compilation	Execution
ACCEL	2816	256
ACCEL2	5120	1024

After compilation, you can redefine MEMORY SIZE to leave only the run-time component in protected memory. This will make more space available during execution for STRINGS, and in the case of ACCEL, for arrays. ACCEL2 has control options which enable you to limit compilation to only that part of the program which is performance critical. This helps you to contain code expansion.

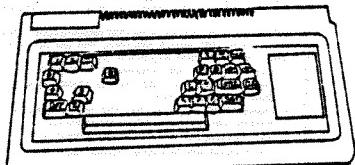
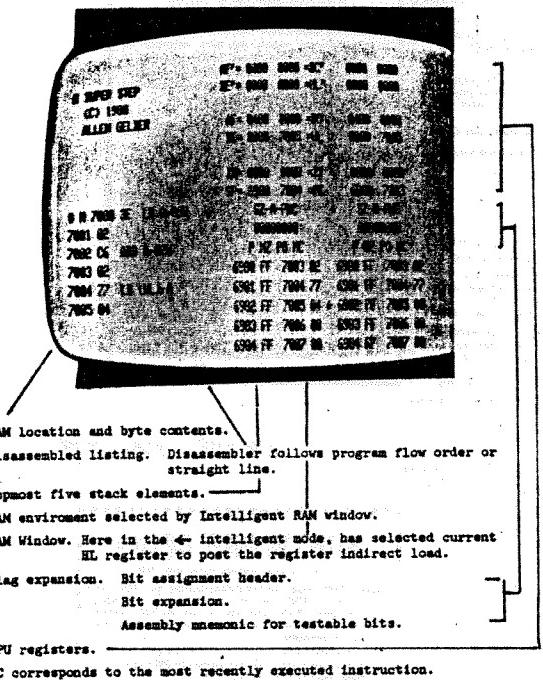


Fig. 2. Implicit keypad under SPRSTP

- CONTROL POINTS:** There are five control access points.
- Control point 1: Open when TBUG # prompt character is displayed. like # R, # N, etc.
 - Control point 2: Open under the TBUG # M command, just after user entry of a two byte RAM address.
M mmm bb
This is the access point shared by the modify memory functions of TBUG. Most SPRSTP keys are accessible here, including SPACEBAR and : TRACE.
 - Control point 3: Open under : TRACE mode to accept speed change, / SKIP, and 2-HALT.
 - Control point 4: Open under (SHIFT) R to accept hex-digit values, ←-left and →-right cursor, and X exit the mode keys.
 - Control point 5: Opens under alternate ← RAM Window mode keystrokes to accept a two byte RAM address or X exit the mode. The value entered defines the user RAM Window environment.

5.



Fig. 1 The \$19.95 Package

Super STEP

Size: 1E00H bytes.
Display: Z80 Model with stack and flags.
Intelligent RAM Window.
Disassembled program listing.
Modes: Single-step and 2-speed TRACE.
Direct or Single-step CALLs and RSTs.
Key functions: Format and service display.
Local editing
Faster tape I/O.
TRACE control.

Over 30 key functions, 16 page booklet of instructions and examples.\$19.95

ALLEN GELDER SOFTWARE
Box 11721 Main Post Office
San Francisco, CA 94101

KEY FUNCTIONS: By row, from the bottom:	
SPACEBAR	CP2 Single-step current instruction.
2	CP3 HALT key under : TRACE.
(SHIFT) <	CP2 Delete byte, move string to FFFF one up.
(SHIFT) >	CP2 Insert byte, move string to FFFF one down.
/	CP2 Suspends Z80 Model activity, makes dis-assembler straight-line.
	CP3 SKIP current instruction under : TRACE.
(SHIFT) :	CP2 Display ASCII equivalent of current byte.
+	CP2 Displays relative location and byte contents.
(SHIFT) +	CP2 Relative Space memory advance.
3	CP1 Brings up copyright, links TBUG and SPRSTP.
(SHIFT) L	CP1 # L loads faster tapes made by (SHIFT) P.
:	CP2 Display hex/ASCII 16 character line with checksum, scroll workspace.
(SHIFT) *	CP2 Alternates : key between hex and ASCII.
ENTER	CP2 Advances memory display. (TBUG)
CLEAR	CP1 Clears current scrolling field.
CLEAR	CP2 Clears workspace area.
+	CP2 Backspace memory advance.
(SHIFT) *	CP2 Return to Reference location.
(SHIFT) R	CP2 Changes registers. Opens cursor over AF register. User may enter byte value or advance cursor with ← or →. Exit with X.
(SHIFT) P	CP1 # P punches faster tapes.
#	CP2 Sets the Z80 Models.
←	CP2 Changes RAM Window status.
→	CP4 Cursor right under (SHIFT) R register change.
→	CP2 Alternately suppresses/returns unlabeled Model.
←	CP4 Cursor left under (SHIFT) R register change.
1	CP3 Slow speed under : TRACE.
2	CP3 High speed under : TRACE.
(SHIFT) #	CP1 Loads SPRSTP Models with TBUG register contents, same under Control point 2.
CP2	TRACE until 2-HALT or encountering "6 HALT, CALL/RST status. Alternately single-step or directly execute CALLs and RSTs.
:	CP2 Alternately suppressed Workspace display.
-	CP2 Change scrolling mode from full to reduced, back.
(SHIFT) -	CP1 Delink TBUG and SPRSTP.
BREAK	CP1 Delink TBUG and SPRSTP.

6.

